DeepRouting: A Deep Neural Network Approach for Ticket Routing in Expert Network

Jianglei Han*†
*SAP
Singapore
ray.han@sap.com

Aixin Sun†
†School of Computer Science and Engineering
Nanyang Technological University, Singapore
axsun@ntu.edu.sg

***Abstract*—Ticket routing is a part of software support process, where multiple expert groups are involved in processing incident tickets. The goal of routing is to find an expert group which can resolve a ticket at the initial assignment, or when it needs to be transferred to another group. Matching a ticket to its potential resolver effectively provides significant business value for both service providers and their customers. Previous works used hand-crafted features to train predictive models to automate or assist in routing. One of the findings shows that, the similarity between a ticket and an expert group is prominent in identifying the resolver among other groups. Meanwhile, numerous studies demonstrate the effectiveness of deep neural networks in text similarity modeling problems. In this paper, we propose a multi-view deep neural network solution to jointly learn a relevance score for a ticket-group pair, using both text and routing path information. The text relevance is modeled by a classic deep semantic matching model, while the routing graph representation is embedded using a convolutional graph network. Experimental results show that the proposed approach outperforms baseline models in resolver ranking and assistive routing tasks. Comparative experiments also show that text has higher importance than routing path information.**

***Keywords*—Ticket routing; expert network; deep learning**

I. Introduction

In enterprise software support services, incidents and issues are communicated and recorded using electronic *tickets*. A ticket is created by an end-user or a helpdesk personnel, then processed by experts. For the support organization, resolving a ticket promptly and correctly is a Key Performance Indicator (KPI) to fulfill Service Level Agreement (SLA) with their customers. To achieve this, the right expert needs to be identified for a ticket. This is challenging when the problem description is unclear, or there is a large number of service offerings and human experts.

Commonly, human experts are organized by groups, known as *expert groups*, which form the basic ticket processing units. Multiple expert groups form an *expert network*. In such network, each group is responsible for a subset of problem domains. At any given time, only one group is processing a ticket. For example, assuming group A is processing a ticket τ. If A cannot resolve τ, it must be transferred to another group B in the network, *i.e.,* $A \rightarrow B$. If τ is resolved by B, B is the *resolver group*, *i.e.,* $g_\tau^+ = B$. The sequence of groups which have processed the ticket, from the first to the current processing group or the resolver (for a resolved ticket), is the *routing path* or *routing sequence*. For a new ticket, the routing path is an empty list, *i.e.,* $S = \emptyset$. A routing path's length is 1 for a ticket, if the first assigned group is the resolver. Otherwise, the length is the number of groups in the path.

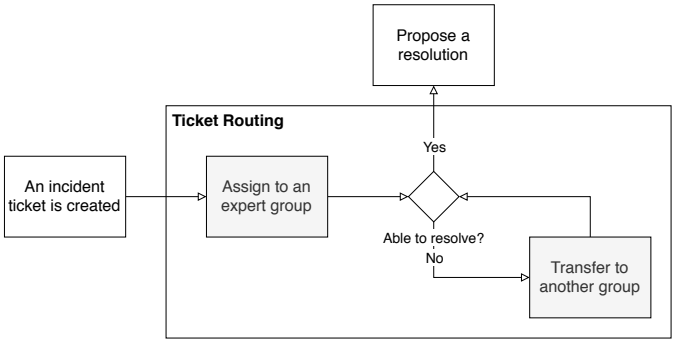


Fig. 1. A high-level overview of ticket processing workflow. Ticket routing includes first group assignment and inter-group ticket transfer.

Ticket routing takes place in two scenarios, as illustrated in Figure 1. Firstly, when the ticket is created and received by the system, the first processing group needs to be assigned. In some systems, this can be manually selected by the end-user during ticket submission. For experienced users, being specific on who should process their tickets is an ease decision. But for new and inexperienced users, making such decision may be challenging. The second scenario happens when the current processing group is not able to resolve the ticket. A next group will be chosen by the current group, depending on their diagnosis of the problem and knowledge about the other groups. Many service providers follow a similar ticket processing workflow in their systems, while some may have some variations, *e.g.,* deploying dispatchers who make centralized routing decisions. In both scenarios, human routing requires considerable level of expertise. The chance of finding the resolver is affected by an individual's previous experience, interpretation of the problem, as well as the familiarity with other expert groups. Routing a ticket to a wrong group would result in longer turnaround time, higher cost, and customer dissatisfaction.

The problem has been increasingly drawing interest from industrial and academic researchers. Focusing on initial group assignment, previous works [1], [2], [3], [4], [5], [6] model

ticket routing as a text classification problem, using ticket text to predict a resolver as target. In a different direction, Shao *et al.* [7], [8] proposed the *EasyTicket* to estimate the probabilities of a group will be selected next, based on the current processing group and routing path, without considering ticket text. Miao *et al.* [9] introduced a generative approach that makes routing decisions based on both ticket text and routing paths. Note that, all above methods assume the first group is already assigned to a ticket by some means. To combine initial group assignment and ticket routing models, Sun *et al.* [10], Xu and He *et al.* [11] proposed two-step frameworks. Both of them used ticket text to select a subset of expert groups as candidate resolvers, then use transfer probabilities for routing in the next step. Lately, Han *et al.* [12] introduced a unified learning-to-rank framework, using hand-crafted features from both ticket and groups. It is able to handle both ticket routing scenarios uniformly (*i.e.,*, initial assignment and ticket routing). Empirical results demonstrate that, among the feature types, ticket-group features that model the similarity between a ticket and an expert group, contribute the most in the proposed routing setup. On the other hand, recent studies show neural network models are effective in modeling text similarities [13], [14], [15], [16], [17], [18]. They are able to generate features automatically from input data, and are able to scale with large number of training instances. Therefore, we are motivated to leverage their strengths and apply them to ticket routing problem.

In this work, we propose DeepRouting, a deep neural network framework for generating feature automatically and computing a relevance score for a ticket and candidate resolver pair. The joint relevance score is a weighted sum of two individual scores, computed in a multi-view architecture. Specifically, in the text-view of both ticket and expert group, we use a classic semantic text matching model to obtain the text relevance. The graph-view outputs expert groups' vector representation in the same size as the text vector. The input to graph-view is pre-trained graph embedding. In such graph, expert groups are nodes, routing path are edges, and ticket text are node features. The relevance scores are computed from both views, and the joint score is used to for candidate resolvers ranking during ticket routing.

In summary, our contributions in this work are the following:

- To the best of our knowledge, DeepRouting is the first work to apply deep neural network models for ticket routing problem, using both text and routing path information.
- This work is also the first attempt to represent an expert network using a graph convolutional network. We use the output embedding for ticket routing.
- Empirically, we show performance comparison between DeepRouting approaches and baseline models.
- Our results show that, among different DeepRouting settings, multi-view outperforms the single-view settings.
- Our results also show that, the ticket text similarity is more important than expert graph for ticket routing.

In the remaining of the paper, we discuss related works in ticket routing and multi-view information matching in Section II. Section III details the views and components. Experimental setup and results are presented in Section IV. Lastly, we conclude and discuss future directions in Section V.

II. Related Work

A. Ticket Routing

The pioneering work of Shao *et al.* [7], [8] modeled ticket routing problem using graphical Markov models. In their formulation, each expert group is represented by a node in a graph. Directed edges exist between two groups with previous ticket transfers, with normalized frequencies as weights. For example, the probability of a ticket to be routed from group A to B, *i.e.,* $P(B|A)$ is estimated by the frequency of $A \rightarrow B$ over all tickets originated from A. Furthermore, the authors applied frequent pattern mining technique to find common paths in routing sequences from archived tickets. They showed that, comparing to individual nodes (*e.g.,* $P(C|B)$), using extracted paths collectively as a super-node (*e.g.,* $P(C|\{A, B\})$) improved routing performance, evaluated by Mean Steps To Resolver (MSTR) score. Miao *et al.* [9] proposed generative models to predict the conditional probabilities of each possible transfer to take place given the terms in a ticket. Both models are only applicable after an initial group cannot resolve the ticket. In comparison, our proposed approach is able to handle both initial group assignment and inter-group transfers.

To make a complete routing solution, Sun *et al.* [10] proposed a content-aware model with multiple steps. The authors implemented a filtering step to reduce the number of candidate groups in routing, based on ticket content similarities. Similarly, Xu and He [11] proposed a Two-stage Expert Routing (TER) model, different from [10] only in how the initial group is identified. As a preprocessing step, expert group representation vectors are trained using their solved tickets with randomly sampled negative tickets. The representation method was proposed by Han *et al.* [19], assuming a group only handles tickets with matching skills, and low dimensional group vectors are learned from resolved tickets. In [11], the first step is to determine the initial group, by taking the nearest neighboring groups in a distributed representation space. Thereafter, the probabilistic transition models are used for ticket transfer. Both [10] and [11] used disjoined and decoupled features for initial group assignment and ticket transfer, which could lead to information loss.

Lately, Han *et al.* [12] proposed a unified framwork for ticket routing (UFTR), built on the feature-based learning-to-rank paradigm. When matching an expert group to a ticket, the framework incorporates hand-crafted features from four types of information. The ticket features are generated from the ticket text. The group features are characteristics of the group based on their resolved tickets; the ticket-group features are relevance and interactive features computed based on both ticket and group. Lastly, the group-group features are transition probabilities between the current routing sequence and candidate groups. These features can be computed offline

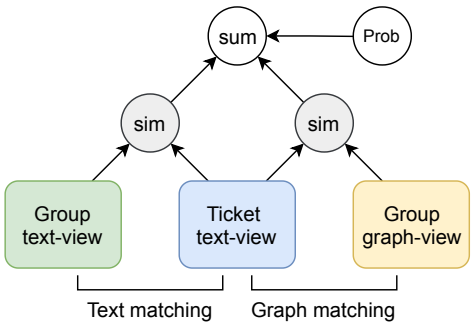


Fig. 2. Multi-view DeepRouting computes a joint relevance score by combining relevence scores from different views.

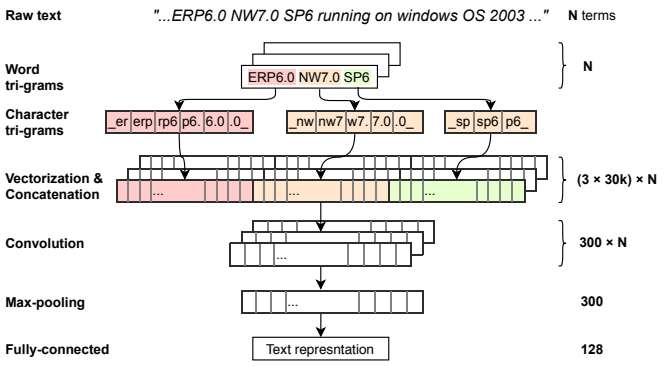


Fig. 3. Overview of text-view processing steps with name labels on the left. Labels on the right indicate the shape of outputs from each step.

and are easily extensible. Compared to previous methods, UFTR applies the same model and steps for both ticket routing scenarios. In this research, we aim to utilize both text and routing information available, leveraging neural network models for automatic feature extraction.

There are also works [20], [21], [22] dedicated to analysis of the contributing factors in routing activities. Particularly, in [21], the authors concluded that the main factors for routing decision-making are individual group's expertise and awareness of others' profiles. While the former is relatively straightforward to obtain, the latter is ambiguous in our problem setting. In another work, Ma *et al.* [23] showed the existence of a theoretical shortest path when applying decentralized search to ticket routing, which is useful in profiling the overall network performance. However, the proposed approach requires the problem areas and difficulty level of a ticket to be known, making it inapplicable to us.

B. Multi-view deep neural network model

Deep neural network models have been increasingly sophisticated and popular for many applications in Natural Language Processing (NLP). Without any expert network structure, ticket routing can be considered as a text matching problem in information retrieval. Deep Structured Semantic Model (DSSM) [13] is a text matching network structure, generating hidden representations for two inputs in identical pipelines. Then, a relevance score (*e.g.,* cosine similarity) between the two representations is computed. DSSM and its variations are widely adopted models for many text matching applications, such as information retrieval [13], [14], [15], recommendation [16], [17], [24], [25], and chatbot [26], [27]. Also working with tickets, Zhou *et al.* [28] proposed a system that matches a ticket to its resolution, using a siamese network architecture, similar to DSSM. Most of these works adopt homogeneous architecture, implementing the same network structures for both inputs. Due to the differences between ticket text and expert network structures, we implement a heterogeneous variation of DSSM.

Mitra *et al.* [18] introduced a duet architecture, using two different deep neural networks to jointly learn local and distributed representation models from a ⟨text query, document⟩ pair for document ranking. The networks for local and distributed models are different. The joint relevant score is the sum of scores from each model. On the other hand, Elkahky *et al.* [17] extended DSSM to a multi-view architecture, for recommending items in online marketplaces. The authors modeled item information from different domains using different networks and train the models jointly. In the multi-view architecture, one of the views (user-view) is shared, while auxiliary items views are modeled individually. In this work, we follow the multi-view architecture in [17] to obtain a joint relevance score for a ticket and a candidate group, score for the resolver should be higher than other groups.

III. System Architecture

DeepRouting architecture is designed for multi-view matching given a ticket and an expert group. As illustrated in Figure 2, it contains two modules for text matching and graph matching, respectively. The text-view takes raw text as input and generates a dense representation vector. Text matching module computes cosine similarity between the vectors for an input ticket and group, respectively. An expert group is represented by a collection of tickets it has resolved. Meanwhile, graph matching part computes the cosine similarity between the ticket text-view and a graph-view vector for a group. The graph-view takes a pretrained graph embedding vector as input and generates a vector with the same dimension as the text-view. Thereafter, the joint relevance score is the weighted sum of scores from both modules. Lastly, the relevance score is multiplied by a group's resolver probability, estimated by the frequency it was the resolver in training set.

A. Text View

We represent each expert group by a collection of tickets it has resolved previously. We follow the deep neural network architecture proposed by [13], [14] for text semantic modeling. The steps and output data shapes from each step are illustrated in Figure 3. Note that, text in tickets are noisy, sparse, and highly domain specific. For this reason, pretrained text embedding methods, such as Word2Vec [29] or GloVe [30] would suffer from high out-of-vocabulary (OOV) issue and poor

performance. In this regard, we employ *word hashing* [13], a character-based tokenization reduces the vocabulary size while preserving the trigram information. We first apply a word-based trigram tokenizer, followed by a character-based trigram tokenizer for each word. The advantage is that, technical jargon and misspellings words are better handled. Then, the vectors of 3 words are concatenated to capture multiword expressions.

Next, features from concatenated vectors are extracted by,

$$\boldsymbol{u}_i = \text{ReLU}(\boldsymbol{W_c^\intercal x_i}), \text{for } i = 1, ..., n \tag{1}$$

where $\boldsymbol{W_c}$ is a linear projection matrix for feature space transformation, shared across all input. ReLU is an activation function, *i.e.,* $\text{ReLU}(x) = \max(0, x)$.

To reduce variable number of feature vectors to a fixed size ticket-level feature vector, we perform dimension-wise max-pooling. Specifically,

$$v_j = \max_{i=1,...,n} \{u_j^i\}, u_j \in \boldsymbol{u} \tag{2}$$

Lastly, a fully-connected layer is applied to extract the high-level semantic embedding,

$$y = \text{ReLU}(\boldsymbol{W_f^\intercal v}) \tag{3}$$

Text-view processes are applied to a ticket and a group text input, and the relevance between them is computed as the cosine similarity,

$$R_{text}(\tau, g) = \text{cosine}(y_\tau, y_g) = \frac{y_\tau^\intercal y_g}{\|y_\tau\| \, \|y_g\|} \tag{4}$$

B. Graph View

Naturally, an expert network has a graph structure and its characteristics. It contains expert groups as nodes and the edges are estimated by inter-group ticket transfers from archived tickets. Some previous works in ticket routing explored such information, using graphical models [8], [7], [22], [9], [11] or extract graph specific features [12] for routing models. However, many of them either considered the graph to be loosely coupled with ticket text, or ignored the text entirely. Moreover, comprehensive graph analytics are costly to compute.

Graph embedding is an effective and efficient method to represent a graph into a low dimensional space, while preserving its information [31], [32], [33]. Different types of graph embedding include node embedding, edge embedding, and subgraph embedding. In our problem setting, we reckon that the first type is applicable. Specifically, node embedding aims to represent graph nodes in low dimensional embedding space, using graph structure information and edge weights. In addition, to incorporate node features as input, we choose Graph Convolutional Network (GCN) [34] over other embedding methods. Specifically, it considers a graph as an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, where n is the total number of nodes. Meanwhile, $\boldsymbol{D}$ is its degree matrix, such that $\boldsymbol{D}_{ii} = \sum_j A_{ij}$. The node features are concatenated in $\boldsymbol{X} \in \mathbb{R}^{n \times m}$, where m is the number of features. To learn a representation vector for each node, GCN propagates through multiple convolutional layers to incorporate neighborhood information in multiple hops. The values of a layer $\boldsymbol{H}^{(l+1)} \in \mathbb{R}^{n \times D}$ is derived as:

$$\boldsymbol{H}^{(l+1)} = \text{ReLU}\left(\tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \boldsymbol{H}^{(l)} \boldsymbol{W}^{(l)}\right), \tag{5}$$

where, $\tilde{\boldsymbol{A}}$ is the sum of $\boldsymbol{A}$ and identity matrix I_N, $W^{(l)}$ is a weight matrix for layer l and $\tilde{D}$ is the diagonal node degree matrix of $\boldsymbol{A}$. $ReLU$ is the activation function introducing non-linearity to the layer output. The parameter values are trained using gradient descent.

In our setting, we consider the expert graph as a homogeneous graph. We use a technical lexicon with 40k terms and phrases as node features. They are automatically generated by semi-supervised learning similar to [35]. After training, each node is represented in a 300-dimensional vector. To compare with the semantic ticket text-view vector, we use a fully-connected layer to project the graph embedding to a vector z_g and compute the cosine similarity between z_g and y_τ.

C. Multi-view DeepRouting

For a pair of ticket and group input, the text matching score and the graph matching score are combined using,

$$R(\tau, g) = \alpha R_{text}(\tau, g) + (1 - \alpha) R_{graph}(\tau, g) \tag{6}$$

where α is a ratio, indicating the significance of text matching. We find the value of α using validation set. During training, the objective is to maximize the likelihood of resolver being identified among non-resolver groups, for across the training tickets. This is equivalent to minimizing the loss function,

$$\mathcal{L} = -\log \prod_{(\tau, g^+)} P\left(g^+|\tau\right) \tag{7}$$

where the probability of the resolver given a ticket is,

$$p\left(g^+|\tau\right) = \frac{\exp\left(\beta_{g^+} R\left(\tau, g^+\right)\right)}{\sum_{g \in \mathcal{G}} \exp(\beta_g R(\tau, g))} \tag{8}$$

where β is a prior probability for a group to be the resolver, estimated from training tickets.

The neural network parameters are trained using Stochastic Gradient Descent with ADADELTA [36] optimizer to dynamically adjust learning rate during training. Each mini-batch consist of 64 positive ticket-group pairs. For each positive pair, 49 negative group are selected, including the non-resolvers in their routing paths and the remaining are randomly sampled. Initially, parameter weights are randomly set. Training of the multi-view network takes more than 50 hours on a server with Nvidia V100 GPU.

IV. Experiment

To evaluate DeepRouting's performance, we conduct two sets of experiments on a proprietary data set. First, we evaluate the effectiveness our model in ranking the true resolver among other groups. Second, we simulate the routing process in Assistive Routing [37], and compare the performance of our

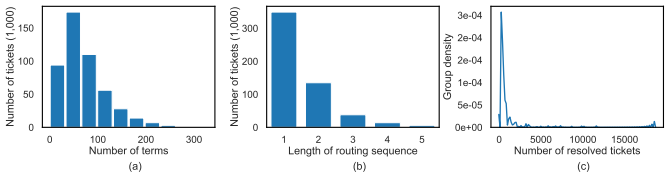


Fig. 4. Ticket length distribution in number of terms, and routing sequence lengths (right) of 500,000 archived tickets.

model with human and algorithmic baselines. Baselines are selected individually for different evaluations.

A. Data set

From an enterprise support system archived, we first collected 500,000 resolved tickets over the past year(s). The overall data statistics are shown in Figure 4. Figure 4(a) shows ticket length distribution in number of terms, showing a peak at around 50 terms, and the majority of tickets are shorter than 150 terms. Figure 4(b) shows that the routing sequence length follows a Power Law distribution. More than half of the tickets are resolved by the first assigned group without further routing. Figure 4(c) is the density distribution of resolver group by the number of tickets they resolved. The majority of groups have solved less than 1,000 tickets during the data collection period.

Next, we select tickets with length between 20 to 150 terms, assuming tickets that are too short or too long are outliers. Among the remaining tickets, we filter again based on the number of tickets their resolver have resolved. Groups that have solved less than 50 or more than 5,000 tickets are removed. From the selected tickets, we randomly select 55,000 tickets for training and 5,000 for validation. For each positive pair, 49 negative group are selected, by including the non-resolvers in its routing path and random sampling for the remaining. For each expert group, 100 tickets are randomly sampled from all tickets it has resolved. From each of routing steps 1 to 4, we sample 500 from the selected tickets for testing. In summary, we have 55,000 tickets for training, 5,000 tickets for validation, and 2000 tickets for test. We preprocessed and manually examined the selected tickets to ensure that any personal information, or any information that could be used to infer user's identity, are removed.

B. Resolver Ranking

During its operation, each step in ticket routing is equivalent to resolver ranking. In the first experiment, we compared the multi-view DeepRouting model with five baseline models for a single step resolver ranking.

BM25 [38] is a classic model for scoring documents with respect to their relevance given a query in information retrieval. We use a term vector to represent each ticket as a query and each expert group as a document, which is represented as a collection of tickets it has resolved in training set.

LDA [39] uses Latent Dirichlet Allocation model for topic modeling. Then, we represent each ticket as a vector in topic space. The number of topics are empirically set to 300.

TABLE I
PERFORMANCE FOR THE RESOLVER RANKING. THE BEST MODEL IS IN BOLD-FACE, WHILE THE SECOND BEST CASE IS UNDERLINED.

	HR@1	@3	@5	@10	MRR
BM25	0.635	0.728	0.773	0.885	0.760
LDA	0.148	0.276	0.503	0.597	0.147
$\text{Word2vec}_{\text{Concatenation}}$	0.407	0.526	0.583	0.733	0.643
$\text{FastText}_{\text{Embedding}}$	0.433	0.597	0.669	0.787	0.670
$\text{UFTR}_{\text{Pointwise}}$	0.687	0.785	0.927	0.948	0.780
$\text{UFTR}_{\text{Pairwise}}$	<u>0.719</u>	<u>0.825</u>	<u>0.942</u>	<u>0.975</u>	<u>0.789</u>
$\text{DeepRouting}_{\text{Text}}$	0.665	0.784	0.895	0.918	0.774
$\text{DeepRouting}_{\text{Graph}}$	0.603	0.742	0.835	0.881	0.754
$\text{DeepRouting}_{\text{MultiView}}$	**0.721**	**0.844**	**0.958**	**0.981**	**0.791**

$\text{Word2vec}_{\text{Concatenation}}$ [29] and **$\text{FastText}_{\text{Embedding}}$** [40] represent a family of word embedding techniques, which are the state-of-the-art for many natural language applications. They use neural network to learn a dense vector representation for each word in the training corpus. Word2vec takes raw text as input, output word vectors with the fixed size. FastText uses sub-word information when learning word representations to handle out-of-vocabulary words. We first trained 300 dimensional word embedding from all tickets in the training set, using both techniques. Subsequently, a ticket vector is the inverse document frequency (IDF) weighted average vector of all its unique word vectors. An expert group vector is represented in the similar way. The relevance score between a ticket and a group is the cosine similarity of their vector representations.

$\text{UFTR}_{\text{Pointwise}}$, $\text{UFTR}_{\text{Pairwise}}$ [12] belong to a ranking framework using multiple types of hand-crafted features. The feature types are Ticket, Group, Ticket-Group, and Group-Group. Note that the original Group-Group features include the transfer probabilities between a current group to a candidate group. Since we only focus on one step resolver ranking in this experiment, only the probability of a candidate being the resolver is used. For pointwise version, a Random Forest Regressor (RFR) model is trained using training set with randomly selected negative samples with 1:1 ratio. During testing, a score is generated for a ticket and each candidate resolver for ranking. The pairwise model is a classification model, determining if a group is likely to rank above the other group. Since we are only interested in resolver's ranking, during testing, we compare the ground truth resolver with each of the other groups to determine its ranking.

$\text{DeepRouting}_{\text{Text}}$, $\text{DeepRouting}_{\text{Graph}}$, $\text{DeepRouting}_{\text{MultiView}}$ are different settings of DeepRouting. Text and Graph are single-view versions, only using text and graph representation of expert groups, respectively. $\text{DeepRouting}_{\text{Text}}$ is effectively identical to the DSSM architecture. Due to the change in network architecture, all models are trained independently on training data, for fair comparison.

Performance for resolver ranking is evaluated by Hit Rate (HR) and Mean Reciprocal Rank (MRR) in a *leave-one-out*

test. Particularly, for each test ticket with a ground truth resolver, 49 other groups are randomly sampled to form the candidate resolver list. Each model outputs a ranking of the resolver in the 50 groups. HR is computed at position 1, 3, 5, and 10. MRR considers the rank of the ground truth resolvers for each test ticket. It's formula is $\mathrm{MRR} = \frac{1}{|\mathcal{T}|}\sum_{i=1}^{\mathcal{T}}\frac{1}{\mathrm{rank}_{resolver}}$.

Shown in Table I, multi-view DeepRouting outperforms the other models across the board. Particularly, it achieves 0.981 in HR@10 and 0.791 in MRR. The second best model is $\mathrm{UFTR}_{\mathrm{Pairwise}}$, which is not far behind. Both top performers incorporate multiple views of information when representing the expert groups. The difference lies in how features are generated. In UFTR, features are manually crafted, while they are automatically generated by neural networks in DeepRouting. Also, both models consider the relative differences of the ground truth and negative groups during training. $\mathrm{UFTR}_{\mathrm{Pairwise}}$ considers the relative position, whereby DeepRouting consider the posterior probability of ground truth group among multiple negative samples in the loss function. Noticeably, BM25 is a strong baseline, performed better than three other baseline models. $\mathrm{FastText}_{\mathrm{Embedding}}$ and $\mathrm{DeepRouting}_{\mathrm{Graph}}$ rely on abstract representation of expert groups from the input, are the least performing models. Input for both could be too abstract to contain valuable information about the expert group.

C. Assistive Routing

Many previous works [8], [10], [9], [11] evaluated their approaches based solely on computed routing results. For example, a system generates the most probable resolver at every step, until the true resolver was found. With the routing path, a MSTR score was computed based on the length of the sequence. In [37], Han and Sun argued that, more information from human routing decision could be utilized in evaluation. Assistive Routing strategy is then proposed. Compared to the fully computed evaluation, it better simulates ticket routing systems in operation, while measuring the performance against the ground truth path, instead of only resolver. Specifically, the system under test generates a list of candidate resolvers at each step. Reviewers in the current group is supposed to determine if the resolver is in the list. Otherwise, they should make their own routing decision.

In this experiment, we assume the top-k candidate groups for a ticket are to be reviewed by a pseudo-reviewer. If the true resolver is in the list, the reviewer could correctly route the ticket to it and the ticket is resolved. Otherwise, the ticket is routed to the next group along the ground truth routing path. The process repeats until the resolver is reached. Each routing step is scored by the distance from its proposed group to the resolver. The best score a test ticket could achieve is 1, when the resolver is proposed in the first step; the score decreases exponentially for every extra step it requires. The overall performance for a test set is the average scores of each test cases, *i.e.,* Mean Average Distance to Resolver (MADR). Formally, $MADR(\mathcal{T}) = \frac{\sum_{t\in\mathcal{T}}\phi(\tau)}{|\mathcal{T}|}$, where $\phi(t) = \frac{\sum_{g\in\hat{S}_\tau}\varphi(g)}{|\hat{S}_\tau|}$, where $\varphi(g) = \frac{1}{2^{\triangle(g,g_{resolver})}}$ is the scoring function for a group g, and $\triangle$ represents "distance between".

In addition to MADR, we also compute the MSTR score for each method. Both scores of DeepRouting are compared with multiple baselines, including the **Human Routing** score, which is computed from the ground truth routing paths in test set. As discussed earlier, most of previous works are partial solution for our problem setting and not directly comparable. Therefore, we choose two complete routing frameworks, Two-stage Expert Recommendation (TER) [11] and Unified Framework for Ticket Routing (UFTR) [12] with different settings as baselines.

TER-FM is a greedy First-order Memoryless Markov model, only considering the current group when determining the next group, *i.e.,*

$$g_{i+1} = \mathrm{argmax}_g P(g|g_i), \forall g \in \mathcal{G}, \tag{9}$$

where g_i is the current group.

TER-FMS is a First-order Multiple active State search model, considering all groups a ticket has been routed to. The next group g_{t+1} is selected using

$$g_{i+1} = \mathrm{argmax}_g P(g|g_r), \forall g_r \in S_\tau, g \in \mathcal{G}\backslash S_\tau, \tag{10}$$

where S_τ is the set of groups that is in the current routing sequence.

TER-VMS is Variable-order Multiple active State search algorithm, using a higher order Markov model. The next group g_{t+1} is selected using

$$g_i = \mathrm{argmax}_g P(g|S_k), \forall g \in \mathcal{G}\backslash S_\tau, S_k \subseteq S_\tau, \tag{11}$$

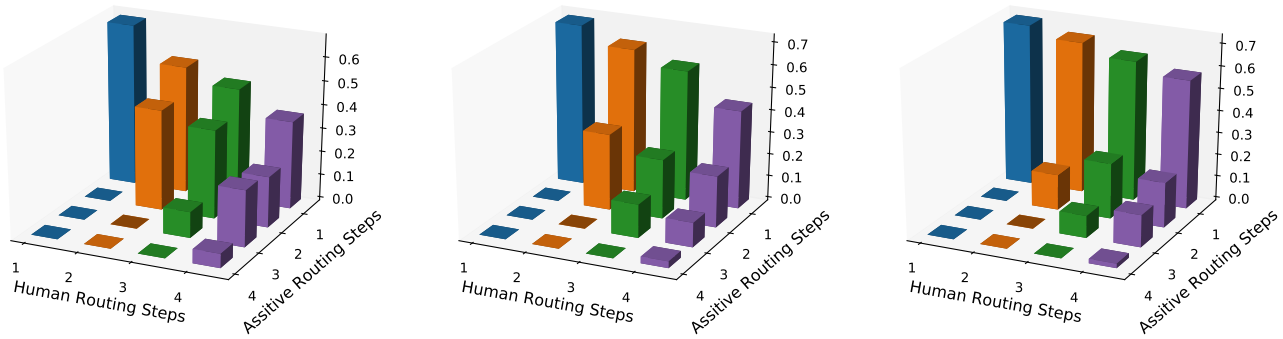
where S_k is a subset of the current routing sequence.

TER-GGT is Generative Greedy Transfer proposed by Miao *et al.* [9]. The routing probability of an expert group g_i to another group g_{i+1} is,

$$P\left(g_{i+1}|\tau, g_i\right) = \frac{P\left(\tau|e_{i,i}\right)P\left(g_{i+1}|g_i\right)}{Z\left(t, g_i\right)} \tag{12}$$

where $e_{i,i+1}$ characterizes the directional edge goes from g_i to g_{i+1}. $P\left(\tau|e_{i,i+1}\right) = \prod_{w_k\in\tau}P\left(w_k|e_{i,i+1}\right)^{\#(w_k,\tau)}$ is the product of probabilities of individual word tokens $w_k \in \tau$ being randomly drawn from all tickets routed along $e_{i,i+1}$. $f(w_k, \tau)$ is an indicator function, equals to 1 if w_k presents in τ; 0 otherwise. $P\left(g_{i+1}|g_i\right)$ is the prior probability of group g_i sending a ticket to g_{i+1}, estimated from archived routing paths. $Z\left(\tau, g_i\right) = \sum_{g_{i+1}\in\mathcal{G}}P\left(\tau|e_{i,i+1}\right)P\left(g_{i+1}|g_i\right)$ is the sum of probabilities to all routing options. Essentially, the formulation in Equation 12 is a softmax function where the edge weights from g_i to all possible nodes are normalized.

$\mathrm{UFTR}_{\mathrm{Pointwise}}$ and **$\mathrm{UFTR}_{\mathrm{Pariwise}}$** are described in Section IV-B. Similar to the settings in Resovler Ranking, we compare three different settings of DeepRouting with the baselines. For



(a) TER-GGT (b) UFTR-Pairwise (c) DeepRouting-DualView

Fig. 5. Number of human routing steps vs. number of Assistive Routing steps in (a) TER-GGT, (b) $\text{UFTR}_{\text{Pairwise}}$, and (c) $\text{DeepRouting}_{\text{MultiView}}$. In each subplot, bars with the same colors represent test tickets that have 1 to 4 steps by human routing. The height of a bar indicates the ratio of tickets that resolved by 1 to 4 steps in Assistive Routing, within its respective color. Note that, we are only interested in assistive routing performance; we exclude cases whereby the resolver is identified after a model failed to recommend after n steps. Therefore, that height of bars with the same color do not necessarily sum up to 1 (best viewed in color).

TABLE II
PERFORMANCE COMPARISON IN ASSISTIVE ROUTING. THE BEST MODEL IS IN BOLD-FACE, WHILE THE SECOND BEST CASE IS UNDERLINED. THE MODELS USE DIFFERENT INFORMATION FOR ROUTING, INDICATED BY † FOR TEXT AND ♯ FOR GRAPH.

	MADR@1	@3	@5	@10	MSTR
Human Routing		0.701			2.500
TER-FM$^{\sharp}$	0.701	0.742	0.754	0.771	2.494
TER-FMS$^{\sharp}$	0.706	0.750	0.781	0.828	2.199
TER-VMS$^{\sharp}$	0.709	0.780	0.804	0.874	1.978
TER-GGT$^{\dagger,\sharp}$	0.711	0.805	0.883	0.924	1.915
$\text{UFTR}^{\dagger,\sharp}_{\text{Pointwise}}$	0.755	0.863	0.913	0.985	1.846
$\text{UFTR}^{\dagger,\sharp}_{\text{Pairwise}}$	<u>0.757</u>	<u>0.893</u>	<u>0.925</u>	<u>0.990</u>	<u>1.676</u>
$\text{DeepRouting}^{\dagger}_{\text{Text}}$	0.719	0.818	0.905	0.958	1.871
$\text{DeepRouting}^{\sharp}_{\text{Graph}}$	0.703	0.774	0.845	0.901	1.971
$\text{DeepRouting}^{\dagger,\sharp}_{\text{MultiView}}$	**0.760**	**0.913**	**0.945**	**0.994**	**1.578**

candidate resolver generation, we use the same information retrieval based with graph expansion method used in [12], for all models. Note that TER-based models hold a fixed candidate set after the first stage, whereas UFTR and DeepRouting models update candidates at every routing step.

The MADR scores at $k \in 1, 3, 5, 10$ and MSTR for assistive routing are shown in Table II. Overall, $\text{DeepRouting}_{\text{MultiView}}$ model outperforms other baselines, improving from human routing baseline by 8% and 40% in MADR@1 and @10, respectively. It scores significantly higher than any of the TER models in MADR. It is also the lowest in MSTR. Among the TER-based models, GGT uses text features, performs better than models using only graph information. The effects of text and graph features are also observable from single view DeepRouting models. Particularly, $\text{DeepRouting}_{\text{Graph}}$ is behind $\text{DeepRouting}_{\text{Text}}$ and barely comparable to the inferior TER models. Even though the graph embedding model considers edge information, the effectiveness in routing is limited. It could be due to the decoupling of graph embedding model training and routing model training. In comparison, the count-based statistical features, though simple, are more expressive and do not require training. Nevertheless, text information has greater weights in solving the ticket routing problem in general. Meanwhile, UFTR-based models are not far from the best DeepRouting model. Even the difference between pointwise and pairwise models are marginal. After all, having both text and graph features at the same time are advantageous over using either one, regardless of the modeling method.

Further, we evaluate the models' effectiveness in solving tickets with various difficulties. We use the number of human routing steps as a proxy for a ticket's difficulty. Particularly, we aim to investigate how the models perform on tickets that took 1 to 4 human routings, only considering top-1 recommendation at each step. They are effectively the fine-grained results from which the MADR@1 results are computed. Among three best model settings, our proposed model (Figure 5) is superior in multiple aspects. Firstly, bars in the farthest row are the tallest in Figure 5(c). This shows that $\text{DeepRouting}_{\text{MultiView}}$ managed to resolve the highest ratio of tickets in the first step. In comparison, $\text{UFTR}_{\text{Pairwise}}$ has the slightly lower ratio, while TER-GGT resolved the least. Secondly, for rows other than the farthest, $\text{DeepRouting}_{\text{MultiView}}$ has the lowest bars compared to the other two models, showing $\text{DeepRouting}_{\text{MultiView}}$ is more likely to propose the resolver in the first Assistive Routing step. Lastly, for tickets that required 3 to 4 human routing steps (green and purple bars), $\text{DeepRouting}_{\text{MultiView}}$ is able to resolve the most, compare to the other methods. As the length of human routing sequence is an implicit indication of a ticket's difficulty, it shows that $\text{DeepRouting}_{\text{MultiView}}$ is able to handle difficult tickets more effectively.

V. Conclusion and Future Work

In this paper, we proposed a deep neural network approach for ticket routing in an expert network. We applied classic text matching and graph embedding in a multi-view architecture to match a ticket with expert groups. The main improvement from the other unified routing framework is that, features are automatic extracted by neural network models. Empirical results show superior performance of the proposed approach in both resolver ranking and Assistive Routing. The results also show that, text similarity is more effective compared to routing graph modeling in ticket routing. A valuable future research direction, is to investigate if simplifying the network structure would improve the overall routing performance. As for the technical choices, we propose to experiment with more text matching frameworks, leveraging both neural networks and hand-crafted features. We would also like to investigate an end-to-end architecture, integrating the graph node embedding during the training process. Lastly, the proposed multi-view framework can be adapted to similar ticket processing systems. However, the generality and robustness of the framework need to be future examined with data from different systems.

Acknowledgment

The first author was in SAP Industrial Ph.D Program, partially funded by the Economic Development Board and the National Research Foundation of Singapore.

References

[1] A. Khan, H. Jamjoom, and J. Sun, "Aim-hi: a framework for request routing in large-scale it global service delivery," *IBM Journal of Research and Development*, vol. 53, no. 6, p. 4, 2009.

[2] H. R. Motahari-Nezhad and C. Bartolini, "Next best step and expert recommendation for collaborative processes in it service management," in *Business Process Management*, 2011, pp. 50–61.

[3] S. Agarwal, R. Sindhgatta, and B. Sengupta, "Smartdispatch: enabling efficient ticket dispatch in an it service environment," in *KDD*, 2012, pp. 1393–1401.

[4] D. Loewenstern and Y. Diao, "A dynamic request dispatching system for IT service management," in *International Conference on Network and Service Management*, 2012, pp. 271–275.

[5] K. Moharreri, J. Ramanathan, and R. Ramnath, "Recommendations for achieving service levels within large-scale resolution service networks," in *ACM India Conference*, 2015, pp. 37–46.

[6] J. Han and M. Akbari, "Vertical Domain Text Classification: Towards Understanding IT Tickets Using Deep Neural Networks," in *AAAI*, 2018, pp. 8202–8203.

[7] Q. Shao, Y. Chen, S. Tao, X. Yan, and N. Anerousis, "Efficient ticket routing by resolution sequence mining," in *KDD*, 2008, pp. 605–613.

[8] ——, "Easyticket: a ticket routing recommendation engine for enterprise problem resolution," *VLDB*, vol. 1, no. 2, pp. 1436–1439, 2008.

[9] G. Miao, L. E. Moser, X. Yan, S. Tao, Y. Chen, and N. Anerousis, "Generative models for ticket resolution in expert networks," in *KDD*, 2010, pp. 733–742.

[10] P. Sun, S. Tao, X. Yan, N. Anerousis, and Y. Chen, "Content-aware resolution sequence mining for ticket routing," in *Business Process Management*, 2010, pp. 243–259.

[11] J. Xu and R. He, "Expert recommendation for trouble ticket routing," *Data & Knowledge Engineering*, vol. 116, pp. 205–218, 2018.

[12] H. Jianglei, L. Jing, and S. Aixin, "UFTR: A Unified Framework for Ticket Routing," *arXiv:2003.00703*, 2020.

[13] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. P. Heck, "Learning deep structured semantic models for web search using clickthrough data," in *CIKM*, 2013, pp. 2333–2338.

[14] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil, "A latent semantic model with convolutional-pooling structure for information retrieval," in *CIKM*, 2014, pp. 101–110.

[15] Z. Dai, C. Xiong, J. Callan, and Z. Liu, "Convolutional neural networks for soft-matching n-grams in ad-hoc search," in *WSDM*, 2018, pp. 126–134.

[16] J. Gao, P. Pantel, M. Gamon, X. He, and L. Deng, "Modeling interestingness with deep neural networks," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014.

[17] A. M. Elkahky, Y. Song, and X. He, "A multi-view deep learning approach for cross domain user modeling in recommendation systems," in *WWW*, 2015, pp. 278–288.

[18] B. Mitra, F. Diaz, and N. Craswell, "Learning to match using local and distributed representations of text for web search," in *International Conference on World Wide Web*, 2017, pp. 1291–1299.

[19] F. Han, S. Tan, H. Sun, M. Srivatsa, D. Cai, and X. Yan, "Distributed representations of expertise," in *SIAM Data Mining*, 2016, pp. 531–539.

[20] Y. Chen, S. Tao, X. Yan, N. Anerousis, and Q. Shao, "Assessing expertise awareness in resolution networks," in *ASONAM*, 2010, pp. 128–135.

[21] G. Miao, S. Tao, W. Cheng, R. Moulic, L. E. Moser, D. Lo, and X. Yan, "Understanding task-driven information flow in collaborative networks," in *WWW*, 2012, pp. 849–858.

[22] H. Sun, M. Srivatsa, S. Tan, Y. Li, L. M. Kaplan, S. Tao, and X. Yan, "Analyzing expert behaviors in collaborative networks," in *KDD*, 2014, pp. 1486–1495.

[23] L. Ma, M. Srivatsa, D. Cansever, X. Yan, S. Kase, and M. Vanni, "Query answering efficiency in expert networks under decentralized search," in *International on Conference on Information and Knowledge Management*, 2016, pp. 2119–2124.

[24] D. Khattar, V. Kumar, V. Varma, and M. Gupta, "Weave&rec: A word embedding based 3-d convolutional network for news recommendation," in *CIKM*, 2018, pp. 1855–1858.

[25] X. Luo, Y. Yang, K. Q. Zhu, Y. Gong, and K. Yang, "Conceptualize and infer user needs in e-commerce," in *CIKM*, 2019, pp. 2517–2525.

[26] H.-Y. Shum, X.-d. He, and D. Li, "From eliza to xiaoice: challenges and opportunities with social chatbots," *Frontiers of Information Technology & Electronic Engineering*, vol. 19, no. 1, pp. 10–26, 2018.

[27] J. Gao, M. Galley, L. Li *et al.*, "Neural approaches to conversational ai," *Foundations and Trends® in Information Retrieval*, vol. 13, no. 2-3, pp. 127–298, 2019.

[28] W. Zhou, W. Xue, R. Baral, Q. Wang, C. Zeng, T. Li, J. Xu, Z. Liu, L. Shwartz, and G. Ya Grabarnik, "Star: A system for ticket analysis and resolution," in *KDD*, 2017, pp. 2181–2190.

[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *arXiv:1301.3781*, 2013.

[30] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[31] H. Cai, V. W. Zheng, and K. C.-C. Chang, "A comprehensive survey of graph embedding: Problems, techniques, and applications," *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, pp. 1616–1637, 2017.

[32] P. Goyal and E. Ferrara, "Graph embedding techniques, applications, and performance: A survey," *Knowledge-Based Systems*, vol. 151, pp. 78–94, 2018.

[33] C. Yang, Z. Liu, D. Zhao, M. Sun, and E. Y. Chang, "Network representation learning with rich text information," in *IJCAI*, 2015.

[34] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *ICLR*, 2017.

[35] J. Han, K. H. Goh, A. Sun, and M. Akbari, "Towards effective extraction and linking of software mentions from user-generated support tickets," in *CIKM*, 2018, pp. 2263–2271.

[36] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv:1212.5701*, 2012.

[37] J. Han and A. Sun, "Mean average distance to resolver: An evaluation metric for ticket routing in expert network," in *ICSME*, 2017, pp. 594–602.

[38] S. Robertson, H. Zaragoza *et al.*, "The probabilistic relevance framework: Bm25 and beyond," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 4, pp. 333–389, 2009.

[39] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of machine Learning research*, vol. 3, pp. 993–1022, 2003.

[40] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *arXiv:1607.04606*, 2016.